Rude Q & A for VS.NET Beta 2

Microsoft Confidential

Visual Studio.NET

Q: How can I get the VS.NET Beta 2?
A: VS.NET Beta 2 will be distributed at Tech Ed and other conferences through the summer and fall. It will be polybagged with several newsstand publications in the September timeframe. It will be orderable on CD and DVD from the Microsoft devstore for a small shipping & handling charge (http://developerstore.com/devstore/).

For MSDN Universal subscribers it will be live Monday evening (June 18) on MSDN Subscriber Downloads, and it will be shipped in the July MSDN Universal (it's not in MSDN Professional--a good opportunity to encourage upgrades to Universal ;-). The July shipment will be mailed the last week of June.

Q: Where can I learn more about VS.NET and how to build apps with it?
A: You can get more information from http://msdn.microsoft.com/vstudio. Additionally MSDN is carrying a number of articles and white papers on .NET. Finally there are the newsgroups on MSNEWS.MICROSOFT.COM. The Beta 2 DVD contains content specific to learning about .NET and VS.NET. Numerous websites have been carrying .NET related content since our Beta 1 last fall.

Q: What are the system requirements for Visual Studio.NET Beta 2? (Hardware and Software)
A:

Beta 2	**Minimum Requirements***	**Recommended Requirements***
Processor	PII, 450 MHz	PIII, 600 MHz
RAM	96MB W2K Professional 192MB W2K Server	128MB W2K Professional 256MB W2K Server
Video Card	800x600, 256 colors	1024x768, Hi-Color
OS	Windows 2000 / XP / NT4.0, all Professional or Server editions	
Disk	500MB on system drive, 2.5GB on installation drive DVD for DVD media distribution CD required for CD media distribution	

* These requirements are for Beta 2 only.

Q: Does VS.NET work on Win9x or WinME? Why not?
A: We are not supporting Win9x or WinME for the development environment. For the release product we plan to support Win9x/ME for code execution but not for design time. This decision was made to ensure that developers could benefit from the most advanced platforms features available in the Visual Studio environment.

Q: Is there any support for this beta?
A: Your beta documentation will describe how to get support and to access the newsgroups.

Q: When are you shipping? Is it late?

We are planning to release Visual Studio.NET in the 2nd half of this year. We've been consistent about our plans to release in 2001 for quite a while, and though we certainly would have liked to release sooner, we have provided very early beta access for anyone who would like to try the product, and are now offering a deployment license for Web applications built on Beta 2.

Q: Should I start building production applications with VS.NET Beta 2?
A: Visual Studio.NET Beta 2 is designed for evaluation and education purposes on test equipment only. Installing on computers upon which you depend for day-to-day operations, either for yourself or for your business, is strongly discouraged. This beta software should only be installed on test machines that can be reformatted if needed.

Although a license for redistribution is available (see next question), creating and deploying production applications is not necessarily recommended, due to the stage of the tool's development.

Q: Can I distribute code I build with VS.NET Beta 2?
A: Yes. However, there is a special EULA that applies. You must either 1) be an approved web host provider or 2) go to the <http://beta.visualstudio.net/> web site and accept the click through "go live" license and follow the posted instructions.

Q: Will there be separate betas for Visual Basic, Visual C++, and the other tools?
A: There won't be separate beta programs for those tools. Since there are so many shared features between the tools in Visual Studio, it makes sense to have a single beta program.

Q: What's the difference between the .NET Framework and VS.NET?
A: The .NET Framework is the platform. Visual Studio.NET is the development tool.

Q: It looks like there's no difference between C# and VB. Should I stop using VB and switch to C#?
A: Visual Basic.NET provides the best upgrade and the fastest route for existing VB developers to target .NET. VB.NET was architected specifically for the VB community – to enable VB developers to use their existing skills to build .NET applications. It provides them with the most powerful, productive version of Visual Basic to build Windows, Web, and mobile applications as well as the next generation of XML Web services. C# was built with the C++ community in mind. It provides an object-oriented language with constructs and syntax that will be immediately familiar to C and C++ developers.

Q: Is there functional parity between Visual Basic .NET and C#?
A: Yes. In 99% of the cases both Visual Basic and C# will be able to implement the same functional items.

Q: I heard that the recent language changes to VB.NET will [break interoperability with other languages / make VB.NET non-CLS compliant / break VB code / etc]?
A: This is not true. The recent language changes to VB.NET provide VB developers with the best of both worlds – semantic consistency with existing VB code and seamless cross-language interoperability with other .NET languages. The changes were based on feedback received from thousands of VB developers worldwide and are a testament to the success of the beta program.

Q: Why did Visual Basic.NET break compatibility with previous versions of VB?
A: There were two options to consider when designing Visual Basic.NET—retrofit the existing code base to run on top of the .NET Framework, or build from the ground up, taking full advantage of the platform. To deliver the features most requested by customers (for example, inheritance, threading), to provide full and uninhibited access to the platform, and to ensure that Visual Basic moves forward into the next generation of Web applications, the right decision was to build from the ground up on the new platform.

Q: Why are you making me rewrite all of my existing VB code for VB.NET?
A: You absolutely don't need to rewrite all of your existing code. There are a number of options for existing code ranging from leaving it in VB6 to upgrading to Visual Basic.NET.

- Upgrading to VB.NET – VB.NET provides an upgrade tool with VB.NET that enables you to upgrade existing VB6 projects to Visual Basic.NET. While the tool is not a 100% solution, it will upgrade between 80-90% of your existing code. Upgrading is a great option when you want to do new development in .NET and take advantage of all of the new features in VB.NET
- Interoperating with VB6 – Microsoft provides a rich library for seamlessly interoperating with existing VB code. This enables you to continue using VB6 code that will not require new development or that is in "maintenance" mode. In addition, interop enables you to incrementally upgrade existing VB applications to VB.NET.

Q: I heard Visual C++ isn't targeting .NET. Is Visual C++ no longer strategic to MS?
A: This is incorrect. Visual C++ can absolutely produce code for the .NET platform. In addition we have extensively improved ATL, MFC and the compiler. We have also introduced ATL Server for producing native code Web Service application.

Q: How do you intend to 'satisfy developer needs' when Visual Studio.NET 7.0 doesn't support 64-bit?
A: Developers wanting to target the Itanium processor in Visual Studio.NET 7.0 may do so using the Itanium's 32-bit emulation mode. In addition, Microsoft is aggressively moving forward to support Itanium natively in future releases of Visual Studio.NET.

Q: Will code I write with this Beta work with subsequent releases?
A: There will likely be some incompatibilities between this Beta and subsequent releases

Q: How do you write for the Visual Studio shell?
A: The IDE now exposes a series of interfaces that you can program directly against with the included VS IDE macro language. Alternatively you can join VSIP (vsip@microsoft.com) for a deeper level of integration.

Q: Where is Visual InterDev?
A: Visual Interdev has been included into the Visual Studio.NET IDE. All languages will have access to the market leading features of the current Visual InterDev product. This enables all of the previously separate programming tools to leverage the many capabilities provided by Interdev, including its HTML editors and remotely hosted project functionality.

Q: Is Visual FoxPro part of Visual Studio.NET?
A: Visual FoxPro 7.0 will not be included in Visual Studio.NET. Because Visual FoxPro 7.0 is not a Microsoft .NET language and does not share the Visual Studio.NET integrated development environment (IDE), Microsoft has decided to ship it ahead of, and separately from, Visual Studio.NET. This will enable FoxPro developers to benefit sooner from the new features of the product and enjoy the productivity enhancements that it provides. Visual FoxPro 7.0 is on track to ship by the end of June.
Visual FoxPro does not take advantage of the Common Language Runtime, however developers can build Visual FoxPro COM components and XML Web services that can be used in applications built with Visual Studio.NET. Components and XML Web services built with Visual Studio.NET can also be used in Visual FoxPro applications.

Q: Is this the last version of Visual FoxPro?
A: NO

Q: Why do we still make Visual FoxPro?
A: 1) We have 250,000+ loyal customers building powerful and interesting applications. 2) It is a great tool for building database applications.

.NET Platform / Runtime

Q. What is Microsoft .NET?
A. Microsoft .NET is Microsoft's framework for creating truly distributed XML Web services that will integrate and collaborate with a range of complementary services to serve customers in ways that today's dotcoms can only dream of. It consists of a well-integrated set of best-of-breed applications that manage the business logic and data storage of these applications.

Microsoft .NET comprises the following:

- Microsoft .NET platform—Includes .NET infrastructure and tools to build and operate a new generation of services; .NET user experience to enable rich clients; .NET building block services, a new generation of highly distributed megaservices; and .NET device software to enable a new breed of smart Internet devices.

- Microsoft .NET products and services—Includes Windows.NET, with a core integrated set of building block services; MSN™ .NET; personal subscription services; Office.NET; Visual Studio® .NET; and bCentral™ for .NET.
- Third-party .NET services—A vast range of partners and developers will have the opportunity to produce corporate and vertical services built on the .NET platform.

Microsoft .NET will take computing and communications far beyond the one-way Web to a rich, collaborative, interactive environment. powered by advanced new software, Microsoft .NET will harness a constellation of applications, services and devices to create a personalized digital experience—one that constantly and automatically adapts itself to your needs and those of your family, home and business. It means a whole new generation of software that will work as an integrated service to help you manage your life and work in the Internet Age.

Q. Why should I use it?

A. The Microsoft .NET platform will revolutionize computing and communications in the first decade of the 21st century by being the first platform that takes full advantage of both.

Microsoft .NET will make computing and communicating simpler and easier than ever. It will spawn a new generation of Internet services, and enable tens of thousands of software developers to create revolutionary new kinds of online services and businesses. It will put you back in control, and enable greater control of your privacy, digital identity and data. And software is what makes it all possible.

Microsoft .NET will only succeed if others share broadly in its success. Microsoft's business philosophy has always been to produce low-cost, high-volume, high-performance software that empowers individual and business users, and creates opportunities for our customers, partners and every independent developer. That philosophy is what sets Microsoft apart from its competitors—and Microsoft .NET takes it to a new level.

Q: What is "managed code"?
A: Managed code is code that runs in the context of Microsoft's Common Language Runtime (CLR). This means that the runtime takes care of many aspects of programming such as memory allocation, garbage collection, thread management

and many other low level details, resulting in safer, more reliable and more robust applications and components. Compilers such as Microsoft's Visual C# and VB.NET produce managed code binaries. The .NET frameworks are written in managed code. Managed code is processor independent.

Q: What happened to (Win)DNA?
A: WinDNA, like J2EE, is a second-generation application architecture. The new .NET Platform is a 3rd generation architecture that enables developers to work in the XML Web services world. In this respect WinDNA has been superceded by .NET.

Q: I heard that the .NET runtime is being ported to other platforms. Does that mean that it's like JAVA - Write once run anywhere?
A: We expect that a number of companies will be interested in implementing the .NET Platform (and the ECMA standard) on many varied devices and platforms. Microsoft has not yet made a decision.

Q: Will there be a cross-platform version of the runtime?
A: We expect that there will be but we aren't prepared to discuss it at this time. Cross-platform has proven to be a nice marketing slogan, but the real excitement is in the interoperability that .NET offers by way of XML Web services.

Q: Is there a separate SDK for compiler writers who want to target IL?
A: No.

Q: How does .NET support my existing COM and COM+ based components?
A: The .NET Framework enables developers to call existing COM and COM+ components as well as enabling .NET components to be exposed to COM and COM+ developers. Additionally a considerable amount of COM+ 1.0 "plumbing," has been simplified in the .NET Framework.

Q: Can and/or should XML Web Services have a UI?
A: No. XML Web services expose (publish) business logic. The developer is left to consume that service and build any UI that is appropriate for a given application.

Q: How do I secure my XML Web services?
A: Since XML Web services look and act externally like Web pages (using HTTP requests and responding with text which happens to be well formed XML), they are protected in much the same way that Web sites are protected: through standard Web server authentication mechanisms and through the use of HTTPS and SSL to protect data on the network.

Q: How does UDDI fit in?
A: It is a Meta discovery service and will assist in the use of XML Web Services. Support is integrated into Visual Studio.NET.

Q: Can Biztalk invoke XML Web services?
A: Yes.

Q: Will Microsoft push SOAP, SCL, and Disco to be W3C standards and how important is this to Microsoft?
A: Yes we will work with those standards bodies to have true open international standards for this. This work, along with our work in ECMA, is very important to us.

Q: Can XSLT be used together with Web Form to generate the code for a Web Form?

A: Yes

Q: How does .NET Framework security compare with ActiveX?
A: The two are vastly different: .NET Framework security is enforced, where ActiveX downloads are based on trust. This means that with ActiveX the user had to choose to trust the code completely, or to not run it at all. The .NET Framework allows code to be partially trusted meaning you authorize it to do only certain things (such as only be able to read from a certain directory) and then no matter what the code does it is not allowed to do more.

Q: How safe is the Beta 2 release of .NET Framework?
A: Like any beta release this one is not of ship quality, including bugs that could allow malicious code to break through the enforcement mechanisms and cause harm.

Q: What is the worst security hole in the Beta 2 release?
A: It is difficult to rank actual bugs, but we can say that no security bug in .NET Framework can weaken existing operating system protections. That is, if you ran malicious managed and unmanaged EXEs, the managed one would not be able to do more harm due to a security bug in .NET Framework. So the large amount of security that this release provides is only making you safer.

Q: Why isn't the Beta 2 release safe?
A: A beta release is never of ship quality, and often some of those bugs will have security implications. Knowing that we will fix many of these bugs in future pre-releases, it is inefficient to do security analysis and testing of early beta products. Please refer to the readme file for more details on this release.

Q: Can I write secure applications on the Beta 2 release?
A: Yes, you can write applications with the Beta 2 SDK that will be secure when run later on version one of .NET Framework. Of course, with the security weakness of Beta 2, any application running on will be subject to the same vulnerabilities.

Q: What is Microsoft doing to ensure that the product release will be secure?
A: Many things. First, we are building in the security architecture from the start because this is essential to achieve a high level of security. When security is involved, both feature design and implementation code are reviewed in addition to extensive testing. Additionally, we are working with industry experts to review the code as well.

Q: The XML in .NET, is it still - like now - based on XSD or will it comply with the W3 schema definitions?
A: We intend to comply with the W3C standard by the RTM of .NET.

Q: I'd like to know XXXX about your submission to ECMA?
A: We're very excited about partnering with HP and Intel in submitting this technology to ECMA. For more details please contact either Tony Goodhew or John Montgomery from Microsoft.

JAVA / J2EE / JUMP

Q: How does Microsoft compete against J2EE?
A: J2EE, and admittedly, Windows DNA, were 2nd generation platforms. Microsoft .NET represents the 3rd generation, which means it a) uses public Internet standards to integrate with many platforms; b) was designed specifically for the Internet to literally make the Internet a programmable platform; and c) embraces the way developers work today, for example, by supporting over 20 different programming languages. J2EE offers none of these benefits.

Q: Will anyone be providing a Java implementation for .NET?
A: Microsoft announced an implementation in January 2001, called JUMP. Java User Migration Path to Microsoft .NET (JUMP to .NET) combines technology and service offerings that enable programmers to preserve, enhance, and migrate Java language projects onto the Microsoft .NET platform. JUMP to .NET enables Microsoft Visual J++® customers and other programmers who use the Java language to take advantage of existing skills and code investments while fully exploiting the Microsoft platform today and into the future. JUMP to .NET provides the easiest transition for Java developers into the world of XML-based Web Services, and dramatically improves the interoperability of the Java language with software written in a variety of other programming languages. More information can be found on http://msdn.microsoft.com/visualj.

Q: But your JUMP support is JDK 1.1.4 only – the real action is on J2, isn't it?
A: Hardly. Our research shows that most Java development is 1.1.4 or earlier, and certainly all of the Visual J++ code is 1.1.4. Our responsibility with JUMP is first and foremost to VJ customers, and we think there's a lot of value for anyone moving from Java to .NET. At the end of the day, it's a platform choice, and we're very confident that .NET will win.

Q: Isn't C# just a knockoff of Java?
A: No. C# is a simple modern component oriented language that is an evolution of C and C++. In developing C#, we address the needs of a certain class of developers who want simplicity and power in a managed environment. Additionally C# innovates in several areas of language design. For example, C# deals with problems like component Versioning and Platform Integration in elegant and innovative ways.

Q: How does .NET Framework security compare with Java?
A: First, .NET Framework security was built-in from the start, not added later. Code access security provides a more powerful and granular infrastructure for security enforcement, with permissions that can be extended by customer code to protect their resources. Evidence-based security policy allows administrators to precisely control what code is assigned what permission. User identity classes unify various forms of authentication into a single model, and customers can add to the model with their own.

Q: What's the difference between the Microsoft Common Language Runtime for .NET and the Java virtual-machine?

A: The .NET Framework is different from Java, down to its fundamental design goals. The .NET Framework was designed to be a high-performance, high-productivity development and execution environment. Java was designed to provide a least-common-denominator set of services across different operating systems. These design differences manifest themselves throughout the architecture.
For example, the .NET Framework is a multi-programming language environment: we have more than 20 language partners who are targeting the .NET Framework. By giving developers a choice of programming languages, we are enabling them to choose the right language for the job at hand or to leverage knowledge of an existing language. Java was designed for a single programming language. The .NET Framework was designed for performance, so it executes native code. Java was designed to be interpreted, and only was it later retrofitted with the capability to compile down to native code.

Q: What are the main differentiators between the SUN/Java initiative and Microsoft.NET?
A: It is never clear what “Java” means anymore. If you were referring to Java the platform, then the .NET Framework technology would be competitive at some high level of abstraction, just like any other environment that includes a set of APIs such as Windows DNA. But the .NET technology is fundamentally different and it is inaccurate to pigeonhole it as a competitive alternative to Java. The most obvious difference is that it can consume multiple programming languages. There are many other equally significant, fundamental differences between the technologies. For example, XML is integral to the platform and is seen in several key technologies including XML Web services.

Mobile / Compact Frameworks / WinCE

Q: What are the .NET Compact Frameworks for WinCE?
A: The .NET Compact Framework is Microsoft's .NET offering for the device developers. The .NET Compact Frameworks will empower developers to create applications and services on devices that richly interact with .NET services that built for desktops, servers and other devices.

Q: Will the .NET Compact Frameworks be offered on other OS's besides WinCE?
A: We have the parts of the .NET Compact Framework running on a non-Microsoft real-time OS, but at this time we can't disclose any details

Q: The .NET Compact Frameworks are "managed code" and C# compiler produces "managed code", does this mean that native code and C/C++ will be going away for embedded developers?
A: No, not at all. There will always be a place for native code applications and components. Code that needs to directly access low level hardware or need exact control of their execution will continue to be written in native code using languages such as C/C++. Native code allows low-level programming and gives full control to the developer. Managed code is intended to make the development process vastly simpler.

Q: What is your time-frame for the .NET Compact Frameworks?
A: We will support .NET development in the next full release of the WinCE operating system.

Q: Can native code interact with managed code running in the .NET Compact Frameworks on WinCE?
A: Yes. The .NET Compact Frameworks on WinCE will allow users to call native code. The experience will be a similar subset to what is available in this regard on the Windows Desktop.

Q: How do .the NET Compact Frameworks relate to the CEF technology available on PocketPC 3.0?
A: CEF (Common Execution Format) is a native code solution that enables a C/C++ application to be compiled into an intermediate format and translated into native instructions when deployed to a device. CEF is supported on PocketPC 3.0 devices. CEF is based on some of the same IL technology that .NET is based on, but the two technologies are separate. CEF = native, un-managed code. .NET = managed code.

Q: Are the .NET Compact Frameworks specific to the C# programming language only?
A: No. The .NET Compact Frameworks utilize the Common Language Runtime which is language neutral. C# is a programming language that was created by Microsoft to allow developers to easily take full advantage of the .NET frameworks and run-time. The .NET Run Time works with IL (IL = Intermediate Language). Any programming language that can produce IL can build .NET applications. On the Windows desktop, multiple languages including C#, VB.NET, and many third party languages including COBOL allow developers to build .NET applications and services. Long term, it is likely that many of these languages will allow .NET development for devices.

Q: How does the .NET Compact Framework differ from the .NET Framework for desktops?

A: In the .NET Compact Framework, we are focusing on meeting the aggressive resource budgets unique to devices. Desktops have comparatively fast processors, lots of ram, virtual memory and hard drives, and unlimited power sources. Most devices do not. .NET development for devices is focused keeping things small and running efficiently on devices. Both the desktop .NET Framework and the device oriented .NET Compact Framework cater to the strengths of the platforms they are intended for.

Q: How big is .NET Compact Framework?
A: This is still under development and is not final. We are optimizing for very small, resource constrained devices.

Q: What is the .NET Compact Framework?
A: There is a version of the .NET Framework called the .NET Compact Framework that enables the fundamental architecture of the .NET Framework on small devices such as mobile phones and digital TVs. The .NET Compact Framework is programming language, hardware, and operating system independent.

Q: What will happen to eVB (VB for WinCE)?
A: Visual Basic and the Visual Basic development community are incredibly important to Microsoft, this includes people using the embedded Visual Basic Tools. BASIC was Microsoft's first product and we will continue to support and evolve Visual Basic. Six Million desktop and device VB/eVB developers are an important to us. eVB3.0 allows developers to write Visual Basic applications that run on Windows CE devices. eVB3.0 is based on script engine technology. Long term, we will migrate eVB developers to VB.NET managed code development. People developing eVB applications today should continue to do so. (We will not discuss what this timeframe is)

Q: When I go from eVB-Script Engine WinCE development to VB.NET development, will I need to change my code?
A: (see question above as well). Yes. We will work to make the transition as easy as possible, but moving from script based VB to VB.NET will very likely need to do some modification of their code. People developing eVB applications today should continue to do so. (We will not discuss the timeframe for supporting VB.NET on devices is)